DREAMER : a Design Rationale Environment for Argumentation, Modeling and Engineering Requirements

Célia Martinie, Philippe Palanque, Marco Winckler, Stéphane Conversy

DREAMER: a Design Rationale Environment for Argumentation, Modeling and Engineering Requirements

Célia Martinie, Philippe Palanque,
Marco Winckler
IRIT – University Paul Sabatier
118, route de Narbonne
31062 Toulouse Cedex 9, France
(+33) 561 556 359
{martinie, palanque, winckler}@irit.fr

Stéphane Conversy
ENAC & IRIT – University Paul Sabatier
7, avenue Edouard Belin
31055 TOULOUSE Cedex
(+33) 562 174 019
stephane.conversy@enac.fr

ABSTRACT

Requirements engineering for interactive systems remains a cumbersome task still under-supported by notations, development processes and tools. Indeed, in the field of HCI, the most common practice is to perform user testing to assess the compatibility between the designed system and its intended user. Other approaches such as scenario-based design promote a design process based on the analysis of the actual use of a technology in order to design new technologies better supporting users' tasks and activities. Some of them also support a critical element in the development of interactive systems: creativity [15]. However, these approaches do not provide any support for a) the definition of a set of requirements that have to be fulfilled by the system under design and b) as a consequence for assessing which of these requirements are actually embedded in the system and which ones have been discarded (traceability and coverage aspects). This paper proposes a tool-supported notation for addressing these problems of traceability and coverage of both requirements and design options during the development process of interactive systems. These elements are additionally integrated within a more global approach aiming at providing notations and tools for supporting a rationalized design of interactive systems following a model-based approach. Our approach combines and extends previous work on rational design and requirements engineering. The current contribution, DREAMER, makes possible to relate design options with both functional and non functional requirements. The approach is illustrated by real size case study from large civil aircraft cockpit applications.

General Terms

Documentation, Design, Human Factors.

Keywords

Design rationale, requirements traceability, user interface design.

1. INTRODUCTION

Traceability of choices and systematic exploration of options is a critical aspect of the development processes in the field of safety critical systems. Some software standards such as DO 178 B [23] (which is widely used in the aeronautical domain) require the use of methods and techniques for systematically exploring design options and for supporting the traceability of design decisions. Similarly, ESARR (Eurocontrol Safety Regulatory Requirement) on Software in Air Traffic Management Systems [8] explicitly requires traceability to be addressed in respect of all software requirements (p. 11 edition 0.2). However, such standards only define what *must* be done in terms of traceability but provide no information on *how* such goals can be reached by analysts and developers. Other approaches such as scenario-based design [9][22] [24] promote a design process based on the analysis of the actual use of a technology in order to design new technologies better supporting users' tasks. Some work such as [15][14] address the aspect of creativity that is of high relevance as far as interactive systems are concerned. However, these approaches provide few support for a) defining the requirements in a way they can be directly associated to every component of the system under design and b) as a consequence, for assessing which of these requirements are embedded in the system and which ones have been discarded during the development process (maybe due to resources constraints, conflicting requirements, …).

Recent work in the field of software engineering has been trying to provide solutions to that problem and a collection of papers on that topic can be found in [7]. One of the remaining problems pointed out by many contributions, such as chapters 1, 19 and 20, is that requirements are poorly or even not addressed. As discussed in [25], this is critical as Requirements Engineering provides input to all the subsequent phases in the development process. This paper addresses the problem of traceability and coverage of requirements in a model-based development process. It addresses the problem by providing an extension to a notation TEAM and its associated tool DREAM which have previously been presented in [12]. The current contribution, DREAMER, makes it possible to relate design options with both functional and non functional requirements. While the approach could address any kind of requirements, we put the emphasis on requirement expressed in standards such as SRS's ARINC [1] and ISO 9126 Software Quality [11].

This paper starts by presenting the basic principles of the TEAM notation and the extensions that have been made to include information related to requirements. Section 3 introduces a case study describing alternative design options for implementing ARINC user interface components. This case study exemplifies how the DREAMER approach supports the design process with respect to requirements providing ways of answering two

SIGDOC'10, September 27–29, 2010, São Carlos-São Paulo, Brazil.

fundamental questions: 1) Which current design (among the many ones available) satisfies a given requirement and 2) What is the exhaustive list of requirements fulfilled by a particular design. Section 4 illustrates all the functions provided by our tool-supported notation for supporting traceability, versioning and collaborative edition of TEAM diagrams. Lastly, section 5 summarizes the finding, draws conclusions and highlights some perspectives to that work.

2. THE TEAM NOTATION in a NUTSHELL

TEAM notation (Traceability, Exploration and Analysis Method) and its CASE tool DREAM (Design Rationale Environment for Argumentation and Modeling) have been originally proposed in [18] to support the systematic exploration of options during the development process of interactive safety critical systems [12]. Hereafter we describe the main concepts of the TEAM notation and the extensions made for tracing requirements.

2.1 The original TEAM notation

TEAM notation is based on Question Option Criteria (QOC) which is design rationale notation introduced by MacLean and al. [13]. QOC notation allows the description of available options for a design question and the selection of an option according to a list of criteria. The TEAM notation is an extension of QOC that enables the structuring and the recording (in an exhaustive manner) of information produced during design meetings. TEAM diagrams cover:

- The questions that have been raised,
- The design options that have been investigated and the ones that have been selected,
- The evaluation performed for the different options,
- The collection of criteria that have been used for evaluating the options considered,
- The collection of factors that have been taken into account and how they relate to criteria,
- The task models corresponding to options,
- The scenarios extracted from the task models that are used to compute, for each option the value of the criteria.

TEAM notation and its associated tool DREAM can leverage the design rationale process for interactive applications by helping engineers in deciding to reuse or not design choices when facing an already experienced issue. Indeed, TEAM diagrams can be considered as very specific design patterns and, as such, can be reused in another design context. Besides this structuring and recording of information, an important feature of TEAM is to record design decisions and relate them to desired quality factors.

Figure 1 illustrates a simple TEAM model aiming at structuring argumentation around the design of the navigation in a list of candidates for a voting system. Supposed that not all the candidates can be displayed in a single window the model represents two options (i.e. circles): the upper one provides scrolling facilities to the users while the lower one proposes a vocal display with navigation commands (previous and next) in that sequence. The triangles on the right hand side of the figure represent a subset of the usability criteria (time to learn, retention over time, error rate … see [11] for a full list) and their connection to the usability factor. The different types of lines between the criteria and options represent the fact that a given option can support (favor) a criterion (the line is bold) or not support it (the line is dotted). For instance, option "provide-scrolling" strongly supports time-to-learn. This is represented by a bold line in the diagram. On the other side, the option "vocal presentation" does not support time-to-learn criterion and is thus represented by a dotted line. TEAM supports more precise connection between elements (including absolute and comparative values) but this is not presented here due to space constraints.

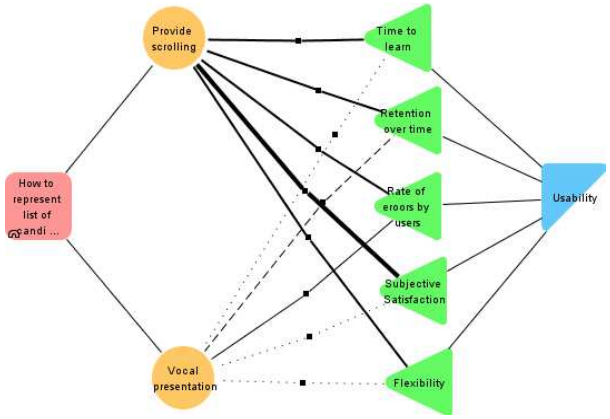


Figure 1. Using TEAM notation to represent relations between criteria and factors in usability

2.2 Adding requirements to notation

Figure 2 provides an exhaustive list of elements in the TEAM notation which also includes all the extensions for supporting requirements. Requirements are depicted as rectangles, questions as rectangles with rounded corners, options as circles, criteria as horizontal triangles and factors as upper part of half a square. Scenarios used to describe a detailed usage of a design option are depicted as squares while arguments (resp. tasks) are depicted as vertical upwards triangles (resp. downwards triangles). The occurrence of any other artifacts used to describe a particular design option (e.g. documents or models specifying the implementation, videos, low-fidelity prototypes, etc) are depicted as paper clip icon and can be attached to any TEAM element.

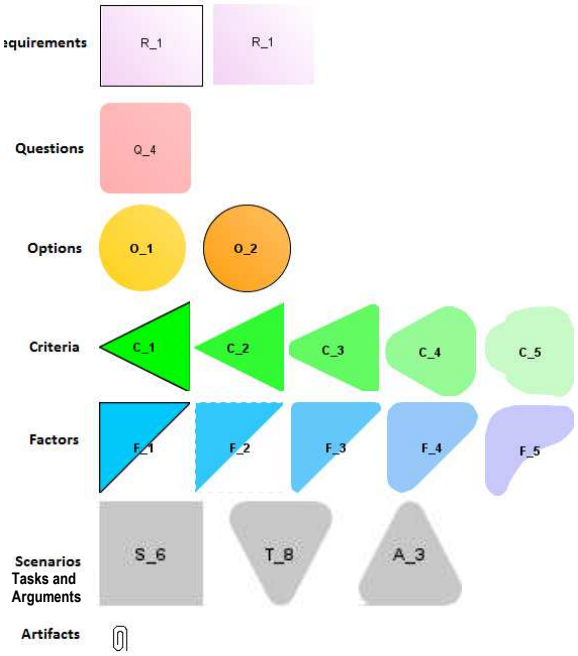


Figure 2. Graphical representation of TEAM notation.

Outlined graphical elements (e.g. the top-left rectangle in Figure 2) represent the fact that, that element has a higher priority than other elements in the design (e.g. top-right rectangle in Figure 2). It is noteworthy that in the case of criteria and factors the graphical representation might also change according to several levels of importance from statured colors and straight shape (very important) to faded colors and irregular shapes (less important). Outlined options (e.g. O_2 in Figure 2) indicate choices made by designers and developers amongst the set of available options.

In its original version, the notation TEAM did not have a representation for requirements. The need for including requirements in rational design diagrams emerged from actual designers whilst trying to determine if the selected options in TEAM diagrams meet functional and non-functional requirements. Indeed, the lack of relationship between design options and requirements prevents designers from exploiting requirements for the generation of options and/or to take into account identified requirements when designing an option. Whilst the integration of requirements represents a small extension to the TEAM notation it has a huge impact on the decision making process based on TEAM diagrams. Indeed, whatever is quality with respect to criteria and factors, a design option might be chosen only on its merit with respect to the coverage of a given critical requirement (such aspect will be detailed in next sections).

3. CASE STUDY

The case study presented in this section is extracted from an industrial cooperation project funded by the DGAC (French civil aviation authority). In this project, one of the main goals was to specify and implement interactive applications in the new generation of interactive cockpits available in small jets but also in large civil aircrafts such as Airbus A380 or Boeing 787.

This project had two main goals:

a) To develop a formal description technique for describing widgets in User Applications for Cockpit Display System; This issue has been addressed by extending the ICO notation [17];
b) To specify User Applications compliant with ARINC 661 standard [1], which is an aeronautical international standard.

Due to space reasons, hereafter we only provide the overall context for the use of rational design approach using a single component from the set of ARINC 661 widgets. In our case study, design options are associated to models describing the actual behavior of user interface widgets used in the Cockpit Display System by means of ICOs.

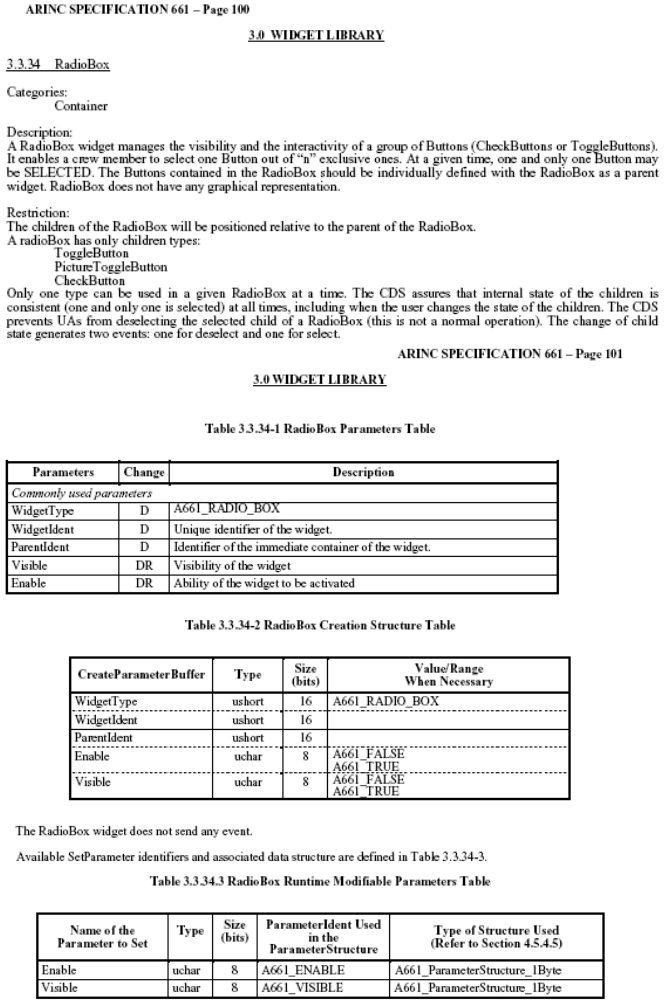
In this section we start by presenting the standard ARINC 661 specification. We then present the context of the case study i.e. the specification of a widget to be used in interactive applications. Section 3.3 presents the formal model of the widget while section 3.4 provides a list of requirements for it. Lastly, section 3.5 presents the design rationale for that widget and its relationship with respect to the identified requirements.

3.1 The standard ARINC 661

The Airlines Electronic Engineering Committee (AEEC) (an international body of airline representatives leading the development of avionics architectures) formed the ARINC 661 Working Group to define the software interfaces to the Cockpit Display System (CDS) used in all types of aircraft installations. The standard is called ARINC 661 - Cockpit Display System Interfaces to User Systems [1].

The CDS (the software system embedded in an aircraft) provides graphical and interactive services to user applications within the flight deck environment. When combined with data from user applications, it displays graphical images and interactive components to the flight deck crew. It also manages user-system interactions by integrating input devices for entering text (via keyboard) and for interacting with these interactive components (via mouse-like input devices). The CDS provides graphical and interactive services to user applications (UA) within the flight deck environment. The communication between the CDS and UAs is based on the identification of user interface components hereafter called widgets. Figure 3 provides a view at glance of the ARINC 661 specification for the widget RadioBox2 (p. 100 and 101). As can be seen on Figure 3, ARINC 661 does not specify the "look and feel" but only the parameters and events.

The next section describes review in the detail all the requirements embedded into the specification of RadioButton2. As the ARINC specification does not impose a particular implementation of user interface widgets that should be embedded into Cockpit Display System (CDS), we have employed DREAMER to document and argument the decisions made on alternative design options.



ARINC SPECIFICATION 661 – Page 100

3.0 WIDGET LIBRARY

3.3.34 RadioBox

Categories:
Container

Description:
A RadioBox widget manages the visibility and the interactivity of a group of Buttons (CheckButtons or ToggleButtons). It enables a crew member to select one Button out of "n" exclusive ones. At a given time, one and only one Button may be SELECTED. The Buttons contained in the RadioBox should be individually defined with the RadioBox as a parent widget. RadioBox does not have any graphical representation.

Restriction:
The children of the RadioBox will be positioned relative to the parent of the RadioBox.
A radioBox has only children types:
ToggleButton
PictureToggleButton
CheckButton
Only one type can be used in a given RadioBox at a time. The CDS assures that internal state of the children is consistent (one and only one is selected) at all times, including when the user changes the state of the children. The CDS prevents UAs from deselecting the selected child of a RadioBox (this is not a normal operation). The change of child state generates two events: one for deselect and one for select.

ARINC SPECIFICATION 661 – Page 101

3.0 WIDGET LIBRARY

Table 3.3.34-1 RadioBox Parameters Table

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_RADIO_BOX
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated

Table 3.3.34-2 RadioBox Creation Structure Table

CreateParameterBuffer	Type	Size (bits)	Value/Range When Necessary
WidgetType	ushort	16	A661_RADIO_BOX
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE

The RadioBox widget does not send any event.

Available SetParameter identifiers and associated data structure are defined in Table 3.3.34-3.

Table 3.3.34.3 RadioBox Runtime Modifiable Parameters Table

Name of the Parameter to Set	Type	Size (bits)	ParameterIdent Used in the ParameterStructure	Type of Structure Used (Refer to Section 4.5.4.5)
Enable	uchar	8	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	8	A661_VISIBLE	A661_ParameterStructure_1Byte

Figure 3. Complete description of RadioBox2 in ARINC 611

3.2 Description of the context

The case study is focused on the design of the ARINC widget RadioBox2 that is used for selecting one button out of several exclusive ones. Even though ARINC 661 specification does not define the look and feel of widgets, examples of such widget are presented in Figure 4.

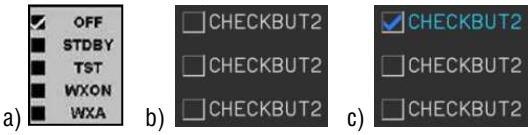


Figure 4. RadioBox2 alternatives for **'look and feel"**

The widget RadioBox2 (circled on Figure 5) is used in several applications embedded into aircraft cockpits such as the Multi Purpose Interactive Application (MPIA) user application (UA). MPIA is a real User Application (UA) aimed at handling several flight parameters. It is made up of 3 pages (called WXR, GCAS and AIRCOND) between which a crew member is allowed to navigate using 3 buttons (as presented at the bottom of each window of Figure 5). WXR page is for managing weather radar information; GCAS is for Ground Anti Collision System parameters while AIRCOND deals with air conditioning settings. Further details about this application can be found in [2].

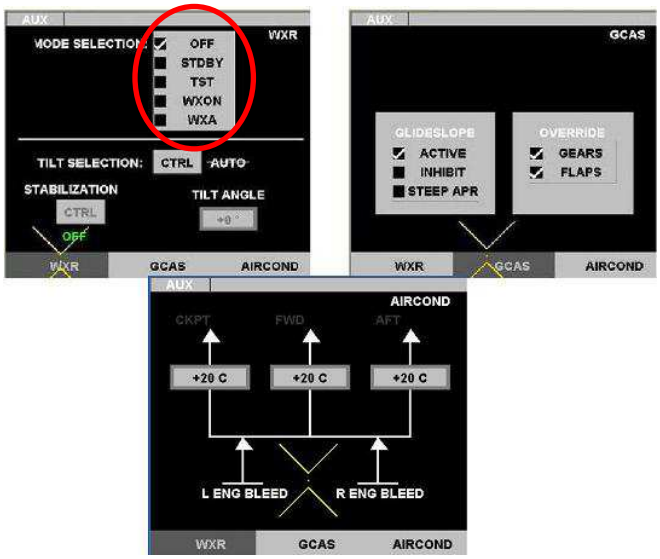


Figure 5. The 3 windows of the user application MPIA

3.3 Formal modeling of ARINC RadioBox2

The use of formal models for describing the behavior is an important requirement to build complex systems such as cockpit display systems. For this reason, the RadioBox2 widget, as well as other widgets contained in the MPIA UA, has been specified using the ICO notation and PetShop tool [20]. Figure 6 provides a view at a glance of the entire ICO models designed to describe the behavior of the ARINC 611 widget RadioBox2.

ICO notation is based on Petri nets. ICO models consist of a set of connected places and transitions; the distribution of tokens among places indicates the availability of actions in the application. Figure 7 shows part of the model of ARINC 661 RadioBox 2, it highlights how to set on/off the visibility of items in the group box widget.

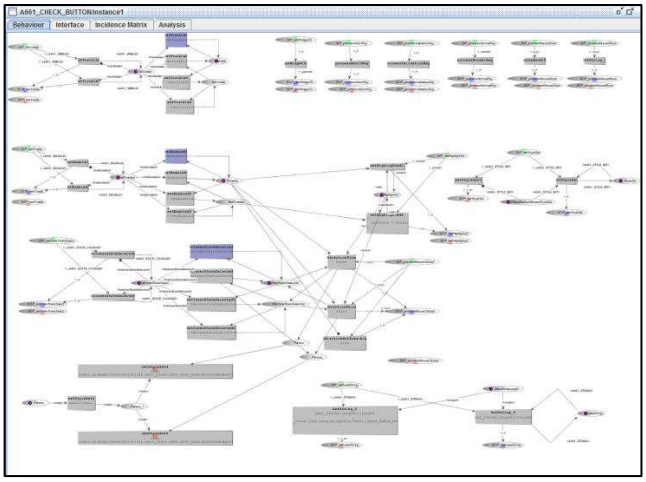


Figure 6. ICO model describing the behavior for the widget ARINC 661 RadioBox2

The details about how to model the behavior of ARINC widgets using ICO is out of the scope of this paper and the interested reader should refer to [16][19] for further information. The main question hereafter is: assuming we need models (ICO models in the present case study) to build complex systems [25]; how we can justify that a given model comply with the requirements (such as ARINC 611)? Other critical and more detailed ones can be derived: Does it comply with all the requirements? Does it comply with only part of them? If so which ones and why some are not taken into account?

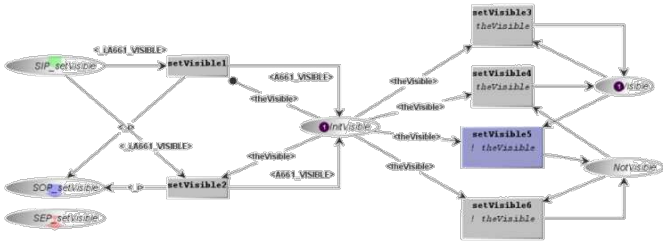


Figure 7. Zoom in ICO model of ARINC 661 RadioBox2 (upper left part of the entire model in Figure 6)

3.4 Requirements for RadioBox2 widget

It is noteworthy that the formal specification of widgets cannot be handled at once and some aspects of the specification are located at different levels of priority according to the phase of the development process (of the project) or to resources availability (such as time, budget and man power for instance).

From the A661 RadioBox2 Domain System Requirement Specification (SRS) the requirements are:

- SRS_SGTK_RB2_DOMAIN_REQ001: RadioBox2 shall be of the Widget library categories: Container.
- SRS_SGTK_RB2_DOMAIN_REQ002: A RadioBox2 shall have only children types: ToggleButton2, PictureToggleButton2, and CheckButton2. Only one type shall be used in a given RadioBox2 at a time. The CDS shall assure that internal state of the children is consistent (one and only one is selected) at all times, including when the user changes the state of the children (the change of child state shall generate two events: one for deselect and one for select).

- SRS_SGTK_RB2_DOMAIN_REQ003: RadioBox2 shall be defined with the parameters as described in the following table:

Parameters	Change	Description
Commonly used parameters		
WidgetType	D	A661_RADIO_BOX2
WidgetIdent	D	Unique identifier of the widget.
ParentIdent	D	Identifier of the immediate container of the widget.
Visible	DR	Visibility of the widget
Enable	DR	Ability of the widget to be activated

- SRS_SGTK_RB2_DOMAIN_REQ004: RadioBox2 shall be created using the parameters defined in the following table:

CreateParameterBuffer	Type	Size (bits)	Value/Range when necessary
WidgetType	ushort	16	A661_RADIO_BOX2
WidgetIdent	ushort	16	
ParentIdent	ushort	16	
Enable	uchar	8	A661_FALSE A661_TRUE
Visible	uchar	8	A661_FALSE A661_TRUE

- SRS_SGTK_RB2_DOMAIN_REQ005: Available SET_PARAMETER identifiers and associated data structure shall be as described in the following table:

Name of the parameter to set	Type	ParameterIdent used in the ParameterStructure	Type of Structure Used
Enable	uchar	A661_ENABLE	A661_ParameterStructure_1Byte
Visible	uchar	A661_VISIBLE	A661_ParameterStructure_1Byte

- SRS_SGTK_RB2_DOMAIN_REQ006: The creation of the RadioBox2 shall be refused if one of the conditions defined in the table below is raised:

Creation error cases	ErrorId
Visible ∉ [A661_TRUE ; A661_FALSE]	CREATE_ABORTED
Enable ∉ [A661_TRUE ; A661_FALSE]	CREATE_ABORTED
Widget hierarchy constraints are not respected	CREATE_ABORTED

- SRS_SGTK_RB2_DOMAIN_REQ007: The RadioBox2 shall send a A661_SET_ABORTED error message with the following identifier ErrorId to UA on SetParameter command if one of the condition defined in the table below is raised:

Run-time error cases	ErrorId
Enable ∉ [A661_TRUE ; A661_FALSE]	A661_OUT_OF_RANGE_ERROR
Visible ∉ [A661_TRUE ; A661_FALSE]	A661_OUT_OF_RANGE_ERROR

- SRS_SGTK_RB2_DOMAIN_REQ008: The RadioBox2 shall be able to change its feel upon reception of an A661 parameter modification command or CDS internal message. For space constraints, we do not present the behavioral requirements in this article.

3.5 Rationalizing the Design of the ARINC RadioBox2

This section describes how we have used DREAMER to support a rationalized and argued design for the complete and unambiguous specification of the widget ARINC 661 RadioBox2. The final DREAMER diagram that is presented in Figure 8 was built according to the following steps:

- First of all, known requirements were added to the diagram. We started with requirement of highest priority for the project that is to have a *Formal description of widgets (using ICO notation)*. Then we added all ARINC 661 requirements for the RadioBox 2 (they have been listed in section 3.4);
- Questions raised by designers during brainstorming meetings were included in the diagram; These questions are explained latter in this section;
- For each question raised, several design options were explored and the relationships between them are depicted by edges in the diagram;
- Any artifacts (i.e. ICO models in the case of the project) related to a particular element were connected to the corresponding design option. The presence of artifacts (if available) is indicated by a paper clip symbol next to the element;
- Prior the selection of design options, criteria and factors influencing the choice were added to the diagram; The selection of criteria and factors has been guided by the ISO 9126 standard on Software Quality [11] as one of the main goals of this project was to address reliability issues for interactive applications in cockpits. For the specification and development of the RadioBox2 widget, three factors have been carefully considered: Reliability, Learnability and Operability (the last two ones being sub-factors of usability in ISO 9126)[1].
- Edges were added for connecting options and criteria. In addition they hold the measures representing the level of compliancy between criteria and options. Lately, edges were connected between requirements and options thus making sure that all requirements were taken into account.

Three main questions have arisen while trying to make unambiguous the behavior of the widget. These questions were driven by two main requirements (i.e. REQ002 and REQ008).

The first question is issued from REQ002 (i.e. *should an option be selected by default?*) and it aimed at deciding the detailed behavior related to the selection of items in the group box. Indeed, the text of SRS specification states that "*The CDS shall assure that internal state of the children is consistent (one and only one is selected) at all times*" but the snapshots given as examples show a state in which no child is selected (see Figure 4b) and another where there at least one child selected (see Figure 4.c). Indeed, this question might lead to two design options: (i) to allow that none can be selected, or (ii) to always have one selected, thus requiring a default selection when none has been selected by the crew.

[1] It is interesting to note that ISO standard 9241 proposes another set of sub-factors for usability namely: Effectiveness, Efficiency and satisfaction. According to that standard, it would have been necessary to identify other criteria for assessing the options.

Three criteria might influence the decision making process on design options: the learning effort, the possibility to manage to use it and the last one concerns the frequency of failure of this widget. The links between options and criteria are given a weight between strongly denied, denied, neutral, supported and strongly supported. For this first question, we can see from Figure 8 that option "0 or 1 selected" is ranked as the best option for the "Learning effort" criterion. In the designers' point of view, it allows a better understanding of the usage of the widget. Then the learnability factor (sub-factor of Usability) that is linked to this criterion could be better supported by this option. The same notation has been performed concerning the operability factor. We can see from the black circle around the second option that it is the one which has been selected. The requirements are filled or not by none or several options. In our case, the second option "0 or 1 selected" is filled in by all the requirements except the second one. The SRS document was presenting a contradiction (that raised this first design question) and the designers have decided rather to select the second option because it should have higher levels of learnability and operability.

In Figure 8 the requirement number 008 is connected to the last two questions which are "How to handle the Focus Management System" and "How to oversee the method call *setParameter*?" Indeed, these questions are directly issued from the definition of the requirement REQ008 concerning the behavior of the widget RadioBox2.

The first question refers to the management of the focus i.e. are the designers going to include the management of focus it in their behavioral description or not? At that time in the project they have chosen not to implement it because they evaluated that the time required to do so was not fitting within the project timeline. Developing the Focus Management using the ICO formalism was not possible in the time frame, even if this solution would have best suited to learnability and operability factors. In this case the reliability factor and formal modeling requirement were considered as more important than the other factors and requirements and this has been captured in the DREAMER model.

The second question is related to the handling of a procedure call that is named *setParameter* and that takes as input parameters the identification of the widget and the state it has to be set to. The SRS document does not indicate how this procedure should handle the *setParamter* when the widget is already in the state that corresponds to the one required by the *setParameter* method call. For instance, this would be the case if the receives a setParamter call to set the selection to the first element of the RadioBox2 widget while it is already selected. The designers chose the second option which only impacts the Reliability factor and that was the more appropriate according to it.

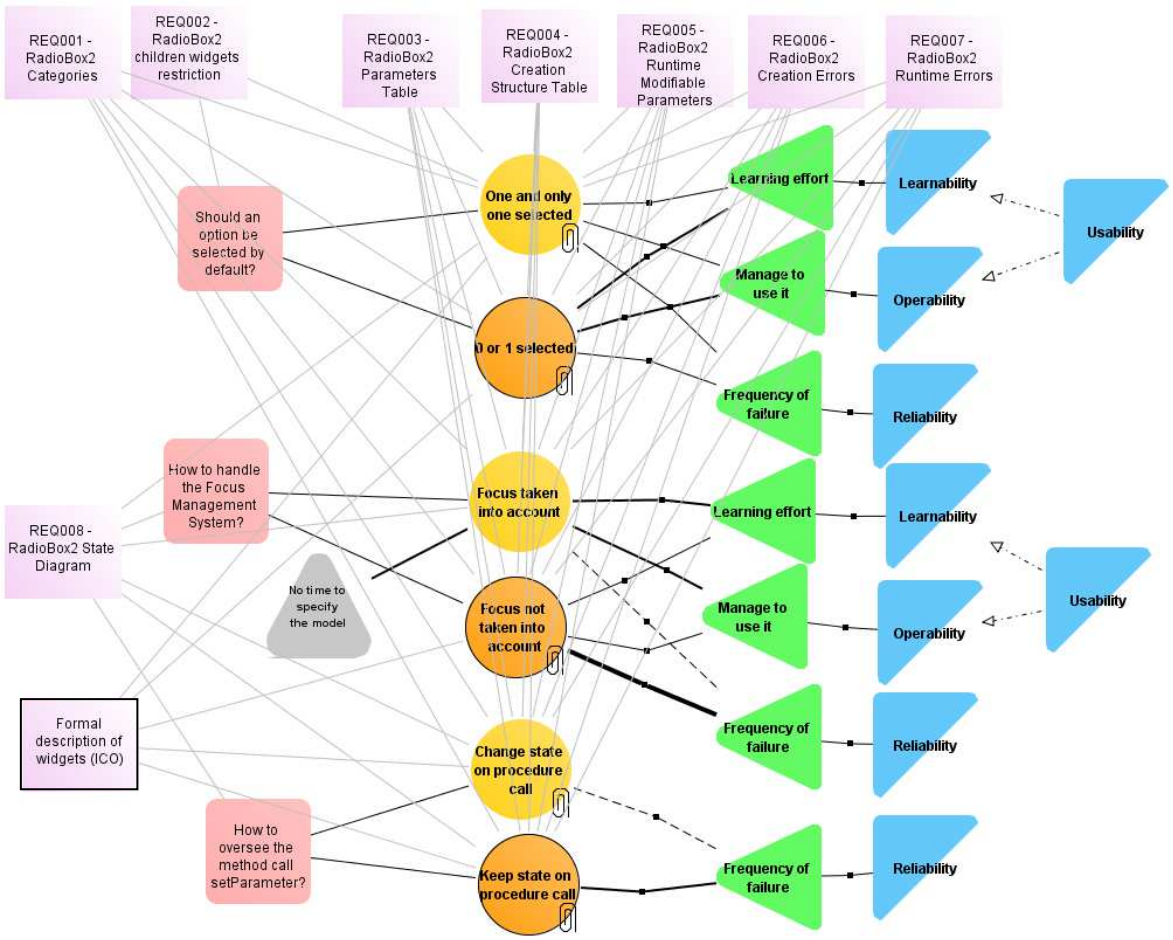


Figure 8. Snapshot of the DREAM diagram for the design of the behavior of the ARINC 661 RadioBox2 widget

4. Tool Support for DREAMER

The diagram presented in Figure 8 was set by the designers using of the DREAMER CASE tool. This CASE tool, which is an extension of DREAM [12], is a software environment to edit, record and analyze TEAM diagrams. The tool is publicly available on the Internet [6]. The first version of the tool, DREAM allows performing the following actions:

- Edit (add, modify and delete) any item from the TEAM notation.
- Connect Question artifacts to Option artifacts.
- Connect Option artifacts to Criterion artifacts and set a weight to this link, depending on how much the option is fulfilling a criterion.
- Connect Criterion artifacts to Factor artifacts.
- Connect models to options as an option can be described by a model (formal or not).
- Connect scenarios to criteria as scenario can be used to evaluate how an option is fulfilling a requirement.
- Attach various types of documents to the different types of TEAM artifacts. They can be related to the design itself or to the project as a whole.

In addition to the functions described above, team work can also be recorded using the two following features:

- Diagram versioning according to the design sessions that took place.
- User roles of the people involved in the design (for instance who decided to select a given option and who was involved in that meeting).

Sometimes diagrams are crowded with a large number of artifacts, and then there is a need to support designers in managing and analyzing the diagram. To this end, a set of visualization tool have been added. For instance, a bifocal view of the diagram has been added to allow focusing on a particular item to explore the various connections it has with the other items in a diagram. Further details about the capabilities of the tool can be found at [12].

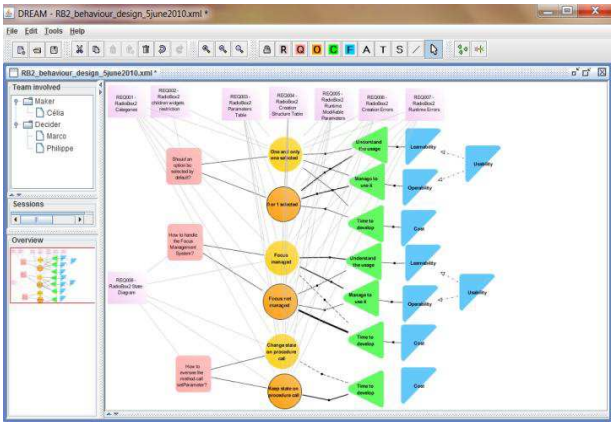


Figure 9. DREAM Software environment

The new version of the tool (see Figure 9) now supports the edition and traceability of requirements within the existing DREAM environment. DREAMER features four main improvements:

- Support for requirements representation;
- Support for relating requirements to design option;
- Visualization of coverage of requirements, options and criteria.

One of the major improvements introduced is the use of visualization techniques for analyzing the coverage of requirements by design options. These visualization techniques, largely inspired from the previous work of Bertin [3] and Henry & Fekete [10], are shown in Figure 10 and Figure 11.

As described in the section 3, REQ001 and REQ003 to REQ007 are fulfilled by all of the design options and then the connecting edges between them and the options make the diagram crowded with edges. To focus on the relationship between design options and requirement they support, Figure 10 shows at a glance which requirements are supported by which options (red color for unsupported requirements and green color for supported requirement). The same kind of view can be used to display the evaluation of an option with regards to criteria (Figure 11). In this colored matrix, the red color corresponds to the strongly denied value and the green color corresponds to the strongly supported value. Between them, a mix of red and green is used to represent the denied, neutral and supported evaluation weights.

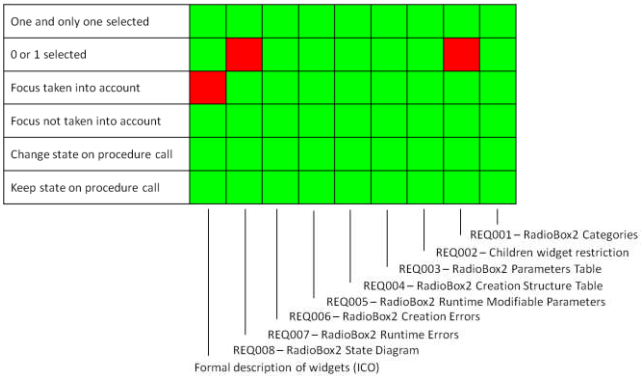


Figure 10. Snapshot of the Colored matrix for requirements traceability with regards to design options

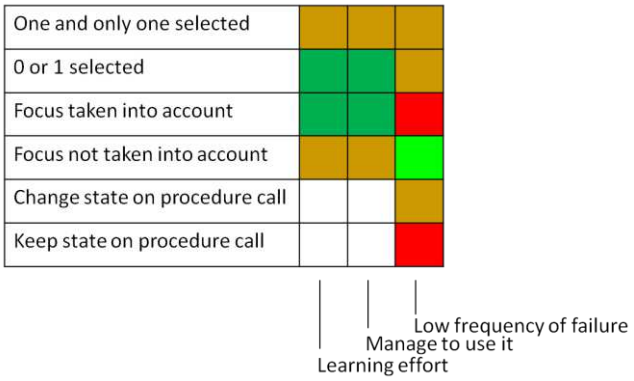


Figure 11. Snapshot of the Colored matrix to visualize the evaluation relationship between options and criteria

It is important to note that these visualizations are embedded into the DREAMER CASE tool so that representations can be automatically generated from TEAM diagrams and interactively manipulated by the designers/developers. They provide terrific support to designers as a design rationale approach is only needed for large and complex systems typically ending up in large and cumbersome diagrams.

5. CONCLUSIONS AND PERSPECTIVES

In this paper we have discussed the problem of traceability of requirements for model-based approaches. It tackles the problem by providing an extension to a notation TEAM and its associated tool DREAM [12]. Whilst some recent approaches are able to deal with the traceability of requirements to pieces of software code [4][21] to pieces of models [5] there is no support for augmenting the choices made during the implementation. DREAMER makes it possible to relate design options with functional and non functional requirements. While other approaches such as SCRAM [24] focus on requirements identification, our approach is intended for supporting the traceability of such identified requirements within the design process of interactive systems.

The current paper has been built from experience drawn from a real industrial project dealing with the behavioral specification of widgets compatible with the standard ARINC 661. However, both the notation and the tool could be used fruitfully with other aspects of the design of interactive systems and other phases of the development process.

REFERENCES

[1] ARINC. ARINC 661 specification: Cockpit Display System Interfaces To User Systems, Prepared by Airlines Electronic Engineering Committee, Published by Aeronautical Radio, Inc, April 22, 2002.

[2] Barboni E., Conversy S., Navarre D. & Palanque P. Model-Based Engineering of Widgets, User Applications and Servers Compliant with ARINC 661 Specification. In Proc. of DSVIS 2006, LNCS n°4323, Springer Verlag. pp. 25-38.

[3] Bertin, J. (1967) Sémiologie Graphique - Les diagrammes - les réseaux - les cartes. Gauthier-Villars et Mouton & Cie, Paris. Réédition de 1997, EHESS.

[4] Boulanger, J.-L., Dao, V. Q. Requirements Engineering in a Model-based Methodology for Embedded Automotive Software. In: IEEE International Conference on Research, Innovation and Vision for the Future, 2008, p. 263-268.

[5] Coninx, K., Cuppens, E., De Boeck, J. & Raymaekers, C., 2007, Integrating Support for Usability Evaluation into High Level Interaction Descriptions with NiMMiT . In: Interactive Systems: Design, Specification, and Verification. 2007, Springr Verlag LNCS, pp. 95-108.

[6] DREAM. At: http://ihcs.irit.fr/dream/index.html

[7] Dutoit, A.H., McCall, R., Mistrík, I., Paech, B. (eds.) Rationale Management in Software Engineering: Concepts and Techniques, Rationale Management in Software Engineering. Springer, 432 pages. 2006. ISBN-10: 3540309977.

[8] ESARR 6. EUROCONTROL Safety Regulatory Requirement. Software in ATM Systems. Edition 1.0. http://www.eurocontrol.int/src/public/standard_page/esarr6.html (2003)

[9] Gregoriades, A., Sutcliffe, A. Scenario-Based Assessment of Nonfunctional Requirements. In IEEE Transactins on Software Engineering, vol. 31, N. 5, May 2005. P. 392-408.

[10] Henry N. and Fekete J-D. MatrixExplorer: a Dual-Representation System to Explore Social Networks. IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2006), 12(5), pp. 677-684, 2006.

[11] ISO/IEC 9126-1:2001. Software engineering -- Product quality -- Part 1: Quality model.

[12] Lacaze, X., Palanque, P., Barboni, E., Bastide, R., Navarre, D., From DREAM to Reality : Specificities of Interactive Systems Development With Respect to Rationale Management. In [7], Springer Verlag 2006, pp.155-170.

[13] MacLean, Allan; Young, Richard M.; Bellotti, Victoria M. E., and Moran, Thomas P. Questions, Options, and Criteria: Elements of Design Space Analysis. Lawrence Erlbaum Associates; 1991; 6, pp. 201-250.

[14] Maiden N.A.M. & Robertson S., 'Developing Use Cases and Scenarios in the Requirements Process', Proc. ICSE 2005 26th Int. Conference on Software Engineering, ACM Press.

[15] Maiden N.A.M., Ncube C. & Robertson S., 'Can Requirements Be Creative? Experiences with an Enhanced Air Space Management System', Proc. ICSE 2007 28th Int. Conference on Software Engineering, ACM Press, 632-641.

[16] Navarre D., Palanque P. & Bastide R. A Formal Description Technique for the Behavioural Description of Interactive Applications Compliant with ARINC 661 Specification. HCI-Aero'04 Toulouse, France, 29 September-1st October 2004. CD-ROM proceedings.

[17] Navarre, D., Palanque, P., Ladry, J., and Barboni, E. 2009. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM Trans. Comput.-Hum. Interact. 16, 4 (Nov. 2009), pp. 1-56

[18] Palanque P. & Lacaze X. DREAM-TEAM: A Tool and a Notation Supporting Exploration of Options and Traceability of Choices for Safety Critical Interactive Systems. In Proceedings of INTERACT 2007, Rio, Brazil, September 2007, LNCS n°4663, Springer Verlag

[19] Palanque P., Ladry J., Navarre D. and Barboni E. High-Fidelity Prototyping of Interactive Systems can be Formal too 13th International Conference on Human-Computer Interaction (HCI International 2009) San Diego, CA, USA.

[20] PetShop: At: http://ihcs.irit.fr/petshop, accessed Jan 2010.

[21] REQTIFY. At: http://www.geensoft.com/en/article/reqtify

[22] Rosson, M.B. & Carroll, J.M. 2002. Usability Engineering: Scenario-Based Development of Human-Computer Interaction. San Francisco: Morgan Kaufmann.

[23] RTCA. Software Considerations in Airborne Systems and Equipment Certification, DO-178B RTCA, Washington D.C. 1992

[24] Sutcliffe A. & Ryan M. Experience with SCRAM, a SCenario Requirements Analysis Method, in Proceedings of the 3rd Int. Conf. on Requirements Engineering, April 1998, pp. 164-173.

[25] van Lamsweerde, A. Engineering Requirements for System Reliability and Security. In Software System Reliability and Security. NATO Security through Science Series - D: Information and Communication Security, Vol. 9. IOS Press, 2007, 196-238.